Letter-to-phoneme conversion by inference of rewriting rules

Vincent Claveau

IRISA - CNRS, Rennes, France

Vincent.Claveau@irisa.fr

Abstract

Phonetization is a crucial step for oral document processing. In this paper, a new letter-to-phoneme conversion approach is proposed; it is automatic, simple, portable and efficient. It relies on a machine learning technique initially developed for transliteration and translation; the system infers rewriting rules from examples of words with their phonetic representations. This approach is evaluated in the framework of the Pronalsyl Pascal challenge, which includes several datasets on different languages. The obtained results equal or outperform those of the best known systems. Moreover, thanks to the simplicity of our technique, the inference time of our approach is much lower than those of the best performing state-of-the-art systems.

Index Terms: phonetization, inference of rewriting rules, phonemization, grapheme-to-phoneme, Pronalsyl Challenge

1. Introduction

Phonetization is the process that aims at transforming a sequence of words into one or several ways to pronounce it. It is a crucial step for speech processing (speech recognition, speech synthesis, audio indexing...). In the first speech processing tools, phonetization was performed by dictionary-based approaches but they have soon reached their limits; many studies have then tried to develop systems able to handle unknown words. In this context, the most common approach is the letter-to-phoneme one: it consists in guessing the pronunciation of a word from its graphemic form. This is also the approach adopted in this paper, which is also known as grapheme-to-phoneme conversion, or phonemization. In practice, it consists in mapping a word-form (represented as a string of letters) to another string of symbols representing a prototypical way to pronounce the word-form.

In our case, this problem of phonetization comes within a broader task of video indexing relying in part on speech processing. This indexing task is beyond the scope of this paper but it sets important properties. Indeed, in this context, out-of-vocabulary words are numerous (neologisms, proper nouns, acronyms, words from specialized domains...), but these words are often present in written documents associated with the video stream (Web sites, newspapers, electronic TV guides...). Moreover, the words (mostly used as query terms) are to be considered independently. It justifies that we are only interested in single word phonetization; we do not aim at developing a whole phonetization system capable of handling complete sentences, liaisons, etc., though such a system could benefit of our work. Finally, for this single word phonetization task, we want a technique producing high quality results, but which is also fast, automatic and portable in order to be adapted to different languages or to subsets of words or even to different speakers.

The system we propose, called IrisaPhon, meets these conditions. The main idea of this approach is to consider the word phonetization as a transliteration problem. Thus, we adapted a simple machine learning technique first developed for transliteration and translation of biomedical terms [1]. This technique allows us to infer very efficiently rewriting rules from examples, that is in our case, from pairs made up of a word-form and its phonetic representation. Apart from these examples, it uses no external knowledge, which makes this technique very portable.

In the remainder of this paper, we first describe some of the main approaches used in phonetization of words. Then, our technique is detailed in Section 3. In Section 4, we report and discuss different evaluation results. Future work and conclusive remarks are finally brought up in the last section.

2. Related work

Many studies have been carried out on automatic (single word) phonetization. Most of them adopt the letter-to-phoneme conversion paradigm: a phoneme sequence is generated from the character sequence representing the word. In some approaches (e.g. the system LIA_PHON [2]), the transition from a sequence to the other is done through the use of tools (transducers, rewriting rules...) developed by an expert. These manual approaches, not portable, are not further detailed in this paper. The machine-learning based approaches which we are interested in try to overcome the limiting role of the expert by relying on examples of word paired with their phonetic representations.

Given the doubly sequential aspect of the problem, the first common sub-task of most of the machine-learning based approaches is to establish the relations between the two sequences, i.e., between the letters and the phonemes in the example pairs. Many systems only consider 1-1 relations [3, 4] but better results have been obtained by considering many-to-many alignments [5, 6] which better capture the fact that several letters may be represented by one phoneme, and that one letter may be represented by several phonemes. Other processing like syllabification [7] are also sometimes used but their benefits are unclear [8, for a discussion]. Given this alignment, the problem can be seen as a machine learning one: a classifier is inferred to propose the right (group of) phoneme given a (group of) letter (or group of letters) and its context. The learning techniques are for example decision trees [3, 9] or *lazy learning* approaches [10]. The main drawback of these approaches is that they poorly take into account the sequential aspect of the input (letters) and usually do not consider the context in the output (phonemes).

Other approaches focus more directly on these sequential aspects and take into account phonemes that have already been transcribed in order to decide of the current phoneme. For example, this is the case for the HMM [11] or analogy-based techniques [12, 13, *inter alia*]. It has been shown that these approches could yield good results provided that the relations between the two sequences were a many-to-many mapping instead of a 1-1 one [5, 13].

Last, some authors have considered approaches relying on machine learning adapted to handle sequential data. This is the case of the CSInf (*Constraint Satisfaction Inference*) system [14]. Let us also cite the studies of Jaimpojamarn *et al.* [6, 8] which rely on a *many-to-many* letter-to-phoneme alignment and propose different classifiers (modified SVM and HMM) for this phonetization task. These last techniques mixing machine learning and sequential approaches are considered as among the best performing techniques. We come back on their performances in Section 4.

3. Phonetizing as rewriting

As we previously said, our tool IrisaPhon is based on a system of rewriting rule inference initially developed to transliterate biomedical terms. The principles of this system are detailed hereafter; the description of its use in translation of biomedical terms can be found in [1].

To phonetize an unknown word, IrisaPhon applies rewriting rules transforming the letter sequence into one or several phoneme sequence. The most probable phonetization is chosen thanks to a language model. The rules and the language model are learnt from training data, that is lists of words-forms (character strings) paired with their phonetic representations (strings of phonetic symbols). Sub-sections 3.1 and 3.2 describe how the rewriting rules are efficiently inferred, and the use of the language model is described in Sub-section 3.3. Some remarks concerning the positioning of IrisaPhon compared with the existing techniques presented above are given in 3.4.

3.1. Inferring rewriting rules

Inferring rewriting rules from the examples is fairly simple. A list of words paired with their phonetic representations is given as input of the system; two characters (resp. # and $) are added to each word and phonetic representation to indicate the beginning and the end of the sequences. This list of examples is then processed as explained in Algorithm 1. The alignment in the first step is made with DPalign (`http://www.cnts.ua.ac.be/~decadt/?section=dpalign`). This piece of software aligns two sequences by minimizing their edit distance, following the dynamic programming algorithm of Wagner & Fischer [15]. The substitution costs are computed on the whole set of pairs to align, and empty characters (written '_') can be inserted if needed. Thus, it is possible to align grapheme and phoneme even though their alphabets is different. Hereafter, a rewriting rule is noted $X \rightarrow Y$ with X a sub-string of letters and Y a sub-string of phonemes; the word-form in input of an example pair W is written $input(W)$, the phonetic representation in output is written $output(W)$; $input$ and $output$ also respectively refer to the premise and conclusion of a rule. Moreover, $align(x, W)$ represents the sub-string of phonemes aligned with the sub-string x in the training pair W. For each difference between two aligned letters, our algorithm generates the rewriting rule which is considered as the best one according to a score function. Many rules are eligible; let us consider the difference o/ə in the pair #phonolog_y$ / #f_ənɑləʤi$. The rewriting rules o → ə, pho → f_ə, #phono → #f_ənɑ, etc., are possible for example.

Algorithm 1 Inference of rewriting rules

```
1: align pairs at the letter level, output the result in L
2: for all pair W_i in L do
3:     for all position at which the 2 aligned letters differ in W_i do
4:         find the best rule hypothesis r in the search space E
5:         add r to the set of rules R
6:     end for
7: end for
```

The score of a rule is simply computed from the list $\mathcal{L}$: it is the ratio between the number of times the rule can be applied and the number of times the premise of the rule matches a sub-string of a word-form. More formally, it is defined by:
score(r) =

$$\frac{|\{W \in \mathcal{L} \,|\, input(r) \subseteq input(W) \wedge output(r) \subseteq align(input(r), W)\}|}{|\{W \in \mathcal{L} \,|\, input(r) \subseteq input(W)\}|}$$

where $\subseteq$ means the inclusion of character strings (e.g. $abc \subseteq aabca$). Among all the possible rules for this example, the rule that maximizes this score is kept and the algorithm can proceed with the next difference in W_i or the next pair of $\mathcal{L}$.

3.2. Exploring the search space

Searching for the best rule among all the possible ones is a key step in our algorithm. In order to do it very efficiently, a hierarchical relation is defined between the rules of the search spaces. This relation is noted down with the symbol $\succeq$ (if $r_1 \succeq r_2$, then r_1 is said more general than r_2).

Hierarchical relation Let r_1 and r_2 be two rules, then $r_1 \succeq r_2 \Leftrightarrow (input(r_1) \subseteq input(r_2) \wedge output(r_1) \subseteq output(r_2))$.

This relation is reflexive, transitive and anti-symmetric [1, for a proof]; it defines a partial order upon the search space $\mathcal{E}$ which thus can be organized as a lattice. Figure 1 presents an excerpt of such a lattice, built up from the difference o/ə in the alignment #phonolog_y$ / #f_ənɑləʤi$.

In practice, the lattices are explored top-down: the rules are generated on-the-fly with a simple operator which produces, for a given rule, every rule immediately more specific. Let us consider for example the rule $r_1 =$ o→ə in the previous example. This rule is the most general one; it is at the top of the lattice in Figure 1. Our algorithm first computes the score of r_1 and then generates more specific rules by adding a letter and its aligned phoneme symbol from W_i at one extremity of the input and output of r_1. For instance, when considering r_1, it gives: h $input(r_1) \rightarrow$ _ $output(r_1)$ and $input(r_1)$ n → $output(r_1)$ n,

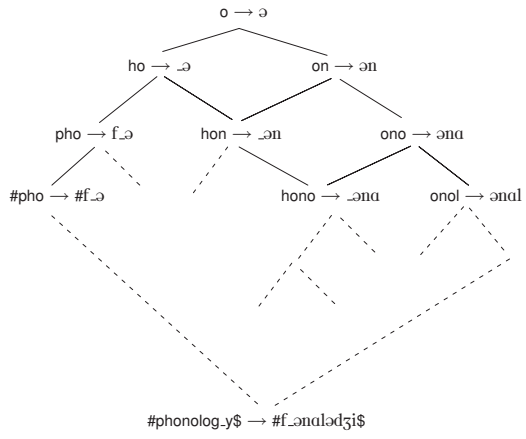

Figure 1: Lattice $\mathcal{E}$ from the example o/ə in #phonolog_y$ / #f_ənɑləʤi$

Dataset	IrisaPhon	MIRA [8]	M-M HMM [6]	Joint n-gram [16]	CSInf [14]	PbA [?]	LIA_PHON [2]
Dutch CELEX	**95.58**	95.32	91.69	–	94.5	–	–
German CELEX	93.60	**93.61**	90.31	92.5	–	–	–
English NETtalk	**71.25**	67.82	59.32	64.6	–	65.35	–
English CMUDict	**74.40**	71.99	65.38	–	–	–	–
French Brulex	**94.75**	94.51	89.77	89.1	–	–	–
French IRISA	**98.60**	–	–	–	–	–	76.8

Table 1: Precision (%) of IrisaPhon and other systems (figures from [8]). Values in bold indicate the best results for a given dataset.

thus ho → ɔ and on → ɔn. One can easily show that the rules r_2 generated are those immediately more specific than r_1, i.e. $\{r_2 \,|\, r_1 \succ r_2 \wedge \nexists\, r_3 \text{ s.t. } r_1 \succ r_3 \succ r_2\}$.

Choosing a score function compatible with this specializing operator (and the lattice structure it implies) makes it possible to quickly explore $\mathcal{E}$. Indeed, computing the score is a time-consuming task since it requires to try to match the rule with every pair of the training set $\mathcal{L}$. But by relying on the way rules are generated, it is possible to compute the score by examining only a subset of $\mathcal{L}$. Let us consider two rules r_1 and r_2 generated from the same example and such that $r_1 \succeq r_2$. When computing the score of r_2, we know that, for any word-form/phonetization pair W, we have: $input(r_1) \subseteq input(r_2) \subseteq input(W)$ (same thing for $output$). It means that to compute the score of r_2, it is sufficient to examine the pairs covered by r_1.

3.3. Choosing the right phonetization

When an unknown word-form needs to be phonetized, every rewriting rule collected in $\mathcal{R}$ are applied; it may lead to numerous phonetization candidates. It is important to note that these phonetizations are aligned with the initial word-form by construction. Among all these candidates, only the most probable will be proposed. This probability is computed in a standard way with a language model applied on the pair word-form/phonetization. Thus, the basic information unit (unigram) is a letter aligned with a phonetic symbol; we note it (for example): $\begin{smallmatrix} \mathsf{s} \\ \mathsf{z} \end{smallmatrix}$. With the usual notation, for a word-form m aligned with one generated phonetic representation f, respectively composed of letters and phonetic symbols (and possibly the empty character _) l_1, l_2, ..., l_m and k_1, k_2, ..., k_m, the probability is computed as in Eqn. 1. In practice, an historic of a few letters is enough and the conditional probabilities are computed on the training pairs $\mathcal{L}$. In the experiments presented below, this historic is set to 6 letters, and a modified Kneiser-Ney smoothing is used.

$$P\begin{pmatrix} m \\ f \end{pmatrix} = \prod_{i=1}^{m} P\left(\begin{array}{c|ccc} l_i & l_1 & ..., & l_{i-1} \\ k_i & k_1 & & k_{i-1} \end{array}\right) \qquad (1)$$

3.4. Positionning of IrisaPhon

Our technique belongs to the family of machine-learning approaches taking care of the doubly sequential aspect of the data. It uses a simple 1-1 alignment, but it is compensated by the fact that there is no a priori limit on the length of the sub-strings of letters/phonemes used in the rules, contrary to other techniques (e.g. MIRA or CSInf). It allows us to better take into account certain long-distance phonetic phenomena and to better disambiguate some character strings.

More generally, this rule based approach can also be seen as a lighter version of analogy-based systems which avoids two of their drawbacks [13]: the combinatory problems induced by handling the very big analogy lattice and the silence (words not transcribed due to the absence of a relevant path in the lattice).

4. Experiments

4.1. Experimental data

In order to evaluate our system, we used several datasets covering different languages and different phoneme sets. We used the data proposed for the *Letter-to-Phoneme Conversion Challenge* (Pronalsyl) of the Pascal network `http://pascallin2.ecs.soton.ac.uk/Challenges/PRONALSYL`. Among the available datasets, we focused on those on which published results existed for comparison purposes. We also used a French lexicon developed in-house and named IRISA hereafter. All these sets are composed of thousands of pairs (16 000 to 300 000) divided into 10 subsets used to perform 10-fold cross-validation.

The performance is measured in terms of word precision averaged on the 10 folds, i.e. the proportion of words perfectly and fully phonetized among the words given to phonetize. Other evaluation measures exist, particularly in context (phonetization of a word inside a sentence, given its part-of-speech...) [17, for a discussion], but do not correspond to our applicative need and would not allow us to compare IrisaPhon to the state-of-the-art.

4.2. Results

Table 1 presents the precision on the different datasets of IrisaPhon, as well as of other state-of-the-art systems when available for comparison purposes. The results are very good: IrisaPhon yields the best precision on almost every set. In particular, it seems robust on difficult datasets (NETtalk and CMUDict), even though a large part for improvement remains. As it is noticed by most studies, for every datasets, most of the remaining errors are due to the presence of import words (words borrowed from another language; mainly English).

As it has been underlined in the introduction, IrisaPhon is to be used in a broader system for indexing video streams in which it may be trained constantly on different datasets. For this reason, the computing time of this training step is worth looking at. Figure 2 details the computing time of the training phase according to the size of the training set (IRISA dataset) on a 2.66 GHz computer with 2 Go RAM running Linux. For comparison, MIRA (which was the best performing system to our knowledge) was reported to have training steps lasting more than 32h for about 56 000 training pairs (with a 2.2 GHz computer) [8]. We also report the precision obtained by IrisaPhon for the different size of training data in Figure 3. In addition, we indicate the optimal precision, that is the precision that would be reached if the good phonetization, when present in the generated candidates, was always chosen. Unsurprisingly, the in-

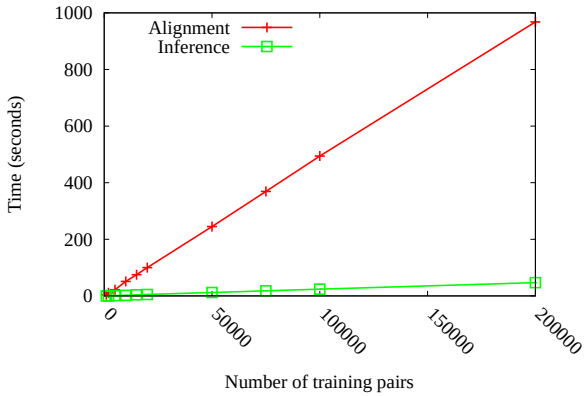

Figure 2: Computing time according to the size of training set

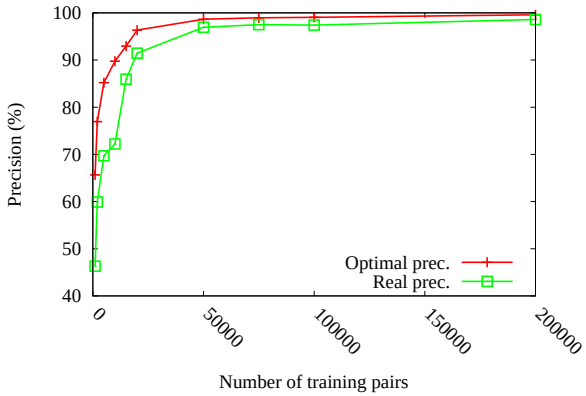

Figure 3: Precision according to the size of training set

ference step, that is the search space exploration, is fairly fast and linear. But it appears clearly that the alignment step is the bottleneck of our technique. It is also worth noting that the performances are lower when the training sets are smaller. Of course, this is due to the smaller number and the lower quality of the inferred rules, but it seems also due to the language model as it is shown by the gap between the real precision and the optimal one.

5. Conclusive remarks and future work

The simple idea of considering the phonetization problem as a transliteration problem yielded interesting results. Compared to state-of-the-art systems, IrisaPhon performs very well in terms of precision and computing time. Of course, the performances are measured on artificially crafted data; thus, the figures obtained must be considered as maxima and an evaluation in real conditions have to be done. Embedding IrisaPhon in our video indexing platform may allow us to address this issue.

Many developments are foreseen for this work. One of them is to lower the cost of the alignment step which masks the gains of our fast inference process. Approximate or greedy alignment methods may address this problem. From a broader point of view, it may be interesting to extend the single word phonetization capabilities of IrisaPhon to a context-sensitive technique which may be useful for other applicative frameworks. For this extension, two different ways are currently explored. First, it is easy to include additional information (like Part-of-Speech or semantic clues) as constraints in the rules and in the language model [1]. Secondly, to take into account the liaison phenomena, it is possible to use a set of special phonetic symbols indicating that a liaison may occur depending on the context (for example the last phoneme of word may change if the next word begins with a vowel phoneme).

6. References

[1] V. Claveau, "Translation of biomedical terms by inferring rewriting rules," in *Information Retrieval in Biomedicine: Natural Language Processing for Knowledge Integration*, V. Prince and M. Roche, Eds. IGI - Global, 2009, ch. 6.

[2] F. Béchet, "LIA_PHON : un système complet de phonétisation de textes," *Traitement Automatique des Langues - TAL*, vol. 42, no. 1, pp. 47–67, 2001.

[3] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter to sound rules," in *Proceedings of the 3rd ESCA Workshop in Speech Synthesis*, Jenolan Caves, Australia, 1998.

[4] R. I. Damper, Y. Marchand, J. D. Marsters, and A. I. Bazin, "Aligning text and phonemes for speech technology applications using an EM-like algorithm," *International Journal of Speech Technology*, vol. 8, no. 2, 2005.

[5] M. Bisani and H. Ney, "Investigations on joint-multigram models for grapheme-to-phoneme conversion," in *Proceedings of the 7th International Conference on Spoken Language Processing*, 2002.

[6] S. Jiampojamarn, G. Kondrak, , and T. Sherif, "Applying many-to-many alignments and Hidden Markov Models to letter-to-phoneme conversion," in *Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, New York, USA, 2007.

[7] Y. Marchand and R. I. Damper, "Can syllabification improve pronunciation by analogy of English?" *Natural Language Engineering*, vol. 13, no. 1, 2006.

[8] S. Jiampojamarn, C. Cherry, and G. Kondrak, "Joint processing and discriminative training for letter-to-phoneme conversion," in *Proceedings of ACL HLT 2008*, Columbus, Ohio, USA, 2008.

[9] W. Daelemans and A. V. D. Bosch, "Language-independent data-oriented grapheme-to-phoneme conversion," in *Progress in Speech Synthesis*, New York, USA, 1997, pp. 77–89.

[10] A. V. D. Bosch and W. Daelemans, "Do not forget: Full memory in memory-based learning of word pronunciation," in *Proceedings of NeMLaP3/CoNLL98*, Sydney, Australia, 1998.

[11] P. Taylor, "Hidden Markov Models for grapheme to phoneme conversion," in *Proceedings of the 9th European Conference on Speech Communication and Technology*, Lisboa, Portugal, 2005.

[12] F. Yvon, "Prononcer par analogie : motivations, formalisations et évaluations," Ph.D. dissertation, ENST, Paris, France, 1996.

[13] Y. Marchand and R. I. Damper, "A multistrategy approach to improving pronunciation by analogy," *Computational Linguistics*, vol. 26, no. 2, 2000.

[14] A. V. D. Bosch and S. Canisius, "Improved morpho-phonological sequence processing with constraint satisfaction inference," in *Proceedings of the 8th Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON'06*, 2006, pp. 41–49.

[15] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 1, pp. 168–173, 1974.

[16] V. Demberg, H. Schmid, , and G. Möhler, "Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, 2007, pp. 96–103.

[17] F. Yvon, P. Boula de Mareuil, C. d'Alessandro, V. Auberge, M. Bagein, G. Bailly, F. Bechet, S. Foukia, J. Goldman, E. Keller, V. Pagel, F. Sannier, J. Veronis, D. O'Shaughnessy, and B. Zellner, "Objective evaluation of grapheme to phoneme conversion for text-to-speech synthesis in french," *Computer Speech and Language. Special Issue on Evaluation*, vol. 12, no. 3, 1998.